MOTOROLA

**AN905**
**Application Note**

# A TRANSPARENT DMA USING A MC6809E MPU AND A MC6844 DMAC

Prepared By:
Tommy Hardy
Systems Applications Engineering Department
Austin, Texas

## INTRODUCTION

This application note describes a system which allows a MC68B09E Microprocessor Unit (MPU) and a MC68B44 Direct Memory Access Controller (DMAC) to individually perform a memory access in one bus cycle.

The memory device (static RAM) is located on a global bus isolated from each master by three-state buffers. These buffers, which are controlled by the MC68B09E clock signal, allow each device access to the global bus for one half of the cycle (MPU E = 0 and DMAC E = 1).

The MPU communicates with a Small Computer Systems Interface (SCSI) bus, which controls a disk storage subsystem, and with a RS-232C bus via a MC68B50 Asynchronous Communications Interface Adapter (ACIA). The DMAC controls high-speed data transfer from the disk storage subsystem on the SCSI bus. In this configuration, the MPU processes data generated by the DMAC while the DMAC is busy generating new tables from disk storage.

## DIRECT MEMORY ACCESS

Direct memory access (DMA) is a technique whereby data may be transferred to or from memory, at very high speed, without MPU intervention. Figure 1 contains a block diagram which illustrates a fundamental system using a DMAC-to-memory configuration. The DMAC provides the means to rapidly move data which the MPU can then manipulate.

Figure 2 illustrates the DMA system discussed in this application note. A MC6809E MPU and a MC68B44 DMAC are used to control a source of data (a hard disk storage subsystem). A MC68B50 ACIA is also used to interface with an RS-232 type terminal.

In many DMA applications using a MC68B09E MPU and a MC68B44 DMAC, the MPU HALT line is asserted (logic 0) by the DMAC to signal that the DMAC is requesting control of the bus. See Figure 3 for a typical DMA example.

In this application, the $\overline{\text{HALT}}$ line is tied high; therefore, the DMAC cannot request use of the bus in the usual manner. Since all operations performed by the DMAC occur on a global bus (shared bus), the MPU need not be halted and thus may address its memory map concurrently with the operation of the DMAC. A memory-mapped technique is used in which the program contained in the MC6809E addresses each portion of the subsystem.

Table 1 defines the memory map. The R/$\overline{\text{W}}$ line is used with the decoded address to select the correct device for the read and write operations. The peripheral chip select control is furnished by the MPU on the A13, A14, and A15 address line inputs to U5, a 1-of-8 decoder.

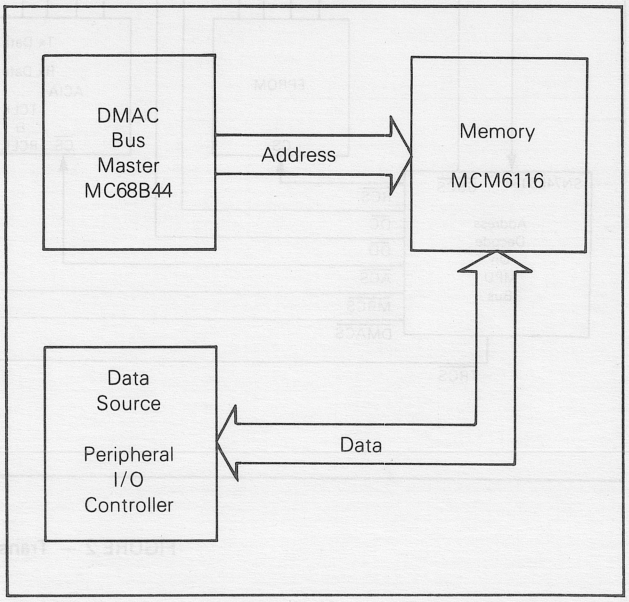


FIGURE 1 — Fundamental DMAC to Memory Configuration

©MOTOROLA INC., 1984

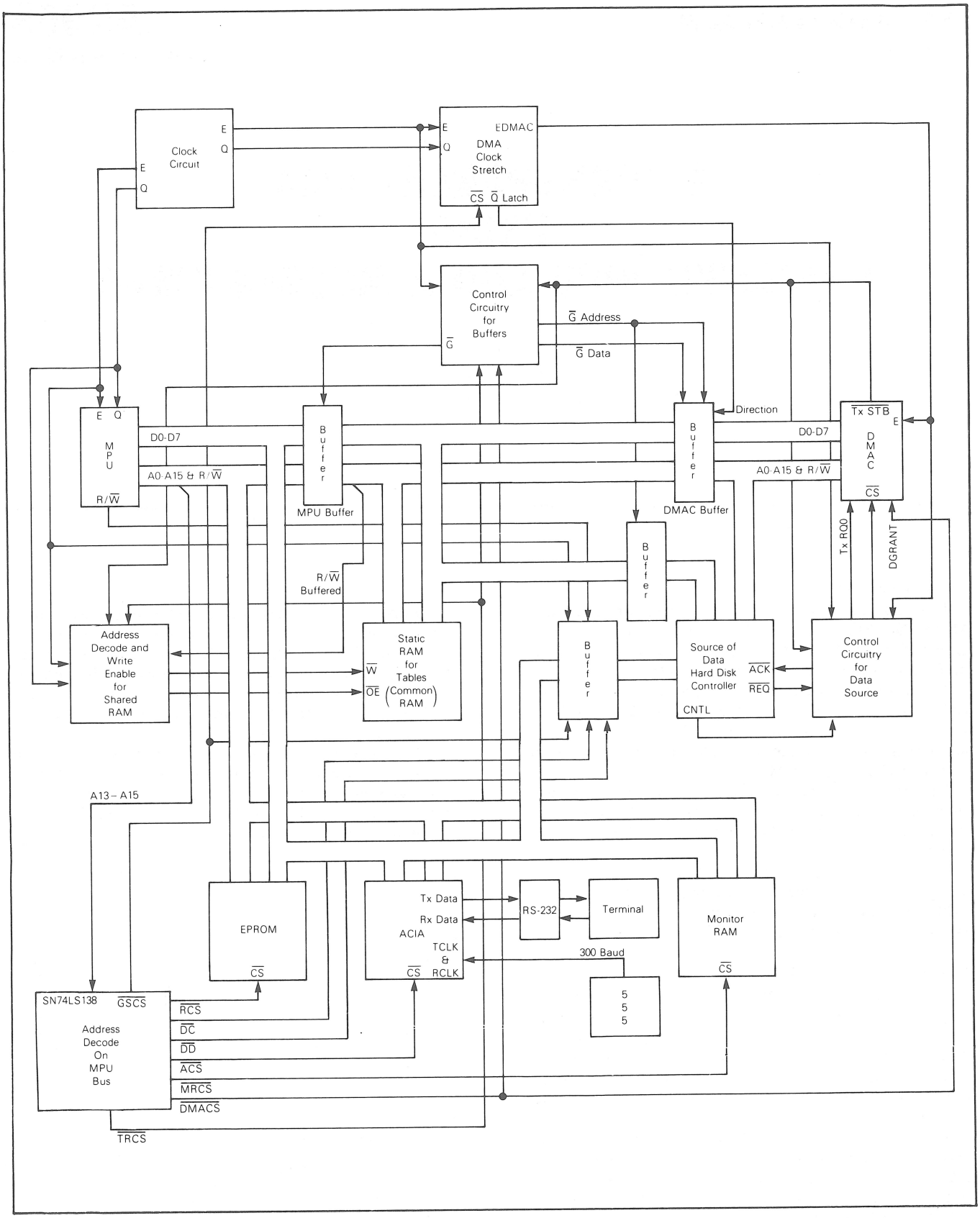


FIGURE 2 — Transparent DMA Block Diagram

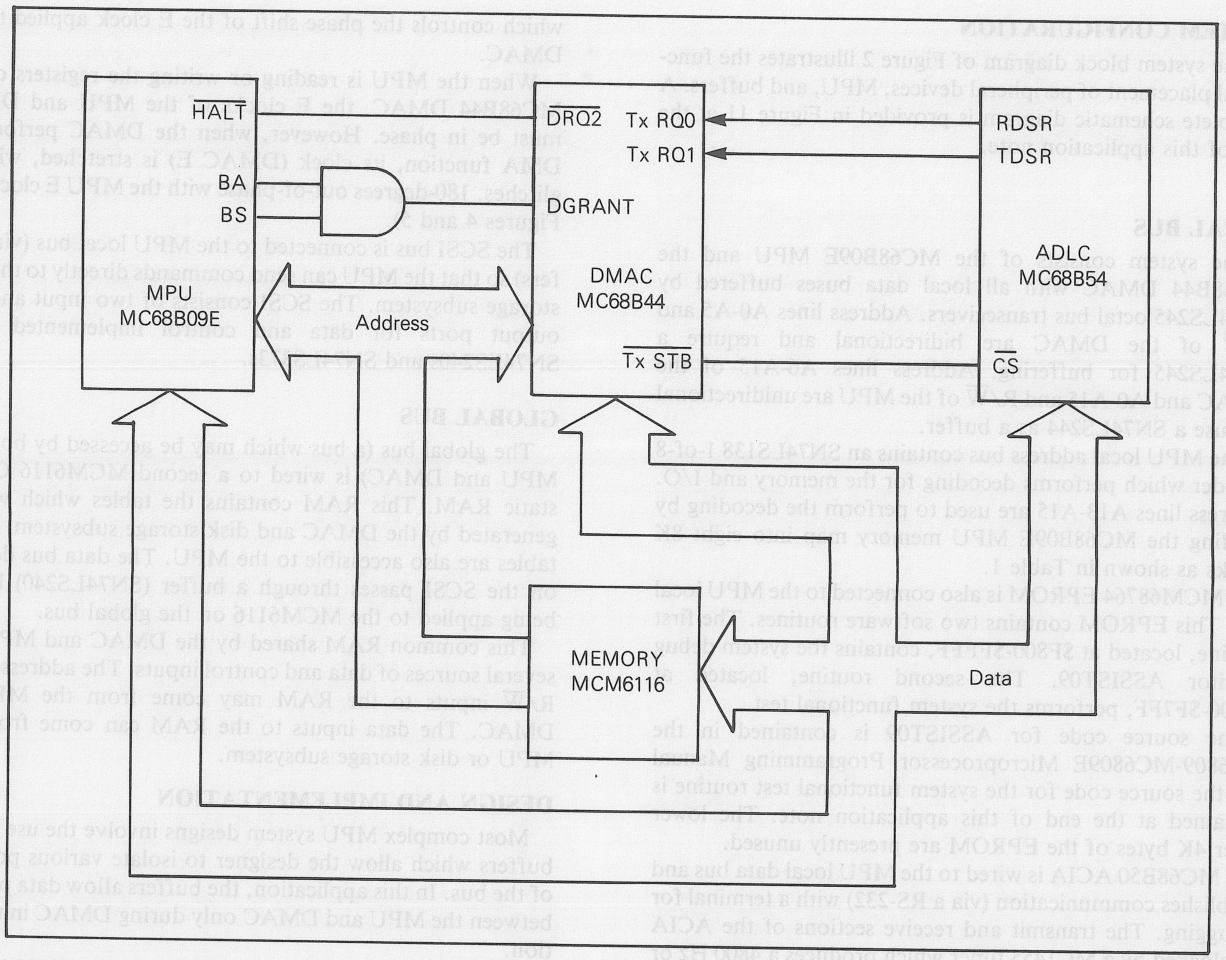


FIGURE 3 — Typical DMA System

TABLE 1 — Memory Map for Transparent DMA

| Device | Function Selected | Address | 1-of-8 Decoder Output |
|---|---|---|---|
| SN74LS373 Buffer or SN74LS240 Latch (U15 & U16) | MPU communicates with SCSI control bus depending on level of the R/$\overline{W}$ line. | $0000-$1FFF | $\overline{DC}$ |
| SN74LS74 Latch (U26) | Stretches the DMAC E clock into and out of the biphase scheme. | $2000-$3FFF | $\overline{CSCS}$ |
| MCM6116 CMOS RAM | Data transfer between MPU and CMOS shared RAM or between SCSI bus and CMOS shared RAM via DMAC. | $4000-$5FFF | $\overline{TRCS}$ |
| MCM6116 CMOS RAM | Data transfer between MPU and DMAC. | $6000-$7FFF | $\overline{MRCS}$ |
| MC68B44 DMAC | Data transfer between MPU and DMAC. | $8000-$9FFF | $\overline{DMACS}$ |
| SN74LS373 Buffer or SN74LS240 Latch (U17 & U18) | MPU communicates with SCSI data bus depending on level of the R/$\overline{W}$ line. | $A000-$BFFF | $\overline{DD}$ |
| MC68B50 ACIA (U9) | MPU communicates with RS-232 | $C000-$DFFF | $\overline{ACS}$ |
| MCM68764 EPROM | MPU reads EPROM. | $E000-$FFFF | $\overline{RCS}$ |

## SYSTEM CONFIGURATION

The system block diagram of Figure 2 illustrates the functional placement of peripheral devices, MPU, and buffers. A complete schematic diagram is provided in Figure 11 at the end of this application note.

## LOCAL BUS

The system consists of the MC68B09E MPU and the MC68B44 DMAC with all local data buses buffered by SN74LS245 octal bus transceivers. Address lines A0-A5 and R/$\overline{W}$ of the DMAC are bidirectional and require a SN74LS245 for buffering. Address lines A6-A15 of the DMAC and A0-A15 and R/$\overline{W}$ of the MPU are unidirectional and use a SN74LS244 as a buffer.

The MPU local address bus contains an SN74LS138 1-of-8 decoder which performs decoding for the memory and I/O. Address lines A13-A15 are used to perform the decoding by dividing the MC68B09E MPU memory map into eight 8K blocks as shown in Table 1.

A MCM68764 EPROM is also connected to the MPU local bus. This EPROM contains two software routines. The first routine, located at $F800-$FFFF, contains the system debug monitor ASSIST09. The second routine, located at $F000-$F7FF, performs the system functional test.

The source code for ASSIST09 is contained in the MC6809-MC6809E Microprocessor Programming Manual and the source code for the system functional test routine is contained at the end of this application note. The lower order 4K bytes of the EPROM are presently unused.

A MC68B50 ACIA is wired to the MPU local data bus and establishes communication (via a RS-232) with a terminal for debugging. The transmit and receive sections of the ACIA are clocked by a MC1455 timer which produces a 4800 Hz or 300 baud signal.

A MCM6116 CMOS static RAM is located on the MPU local bus. It provides stack space for the ASSIST09 debug monitor and allows the user to write small routines. The bus is also set up to address a D-type flip-flop (SN74LS74 latch) which controls the phase shift of the E clock applied to the DMAC.

When the MPU is reading or writing the registers of the MC68B44 DMAC, the E clocks of the MPU and DMAC must be in phase. However, when the DMAC performs a DMA function, its clock (DMAC E) is stretched, without glitches, 180-degrees out-of-phase with the MPU E clock (see Figures 4 and 5).

The SCSI bus is connected to the MPU local bus (via buffers) so that the MPU can send commands directly to the disk storage subsystem. The SCSI consists of two input and two output ports for data and control implemented using SN74LS240s and SN74LS373s.

## GLOBAL BUS

The global bus (a bus which may be accessed by both the MPU and DMAC) is wired to a second MCM6116 CMOS static RAM. This RAM contains the tables which will be generated by the DMAC and disk storage subsystem. These tables are also accessible to the MPU. The data bus defined on the SCSI passes through a buffer (SN74LS240) before being applied to the MCM6116 on the global bus.

This common RAM shared by the DMAC and MPU has several sources of data and control inputs. The addresses and R/$\overline{W}$ inputs to the RAM may come from the MPU or DMAC. The data inputs to the RAM can come from the MPU or disk storage subsystem.

## DESIGN AND IMPLEMENTATION

Most complex MPU system designs involve the use of bus buffers which allow the designer to isolate various portions of the bus. In this application, the buffers allow data passage between the MPU and DMAC only during DMAC initialization.

After the DMAC has been initialized, the MPU buffers turn on when the common RAM is selected and the MPU E clock is high. The DMAC buffers turn on when $\overline{Tx\ STB}$ of the DMAC is active and the DMAC E clock is high (see the truth tables in Figures 6 and the flow chart in Figure 7).

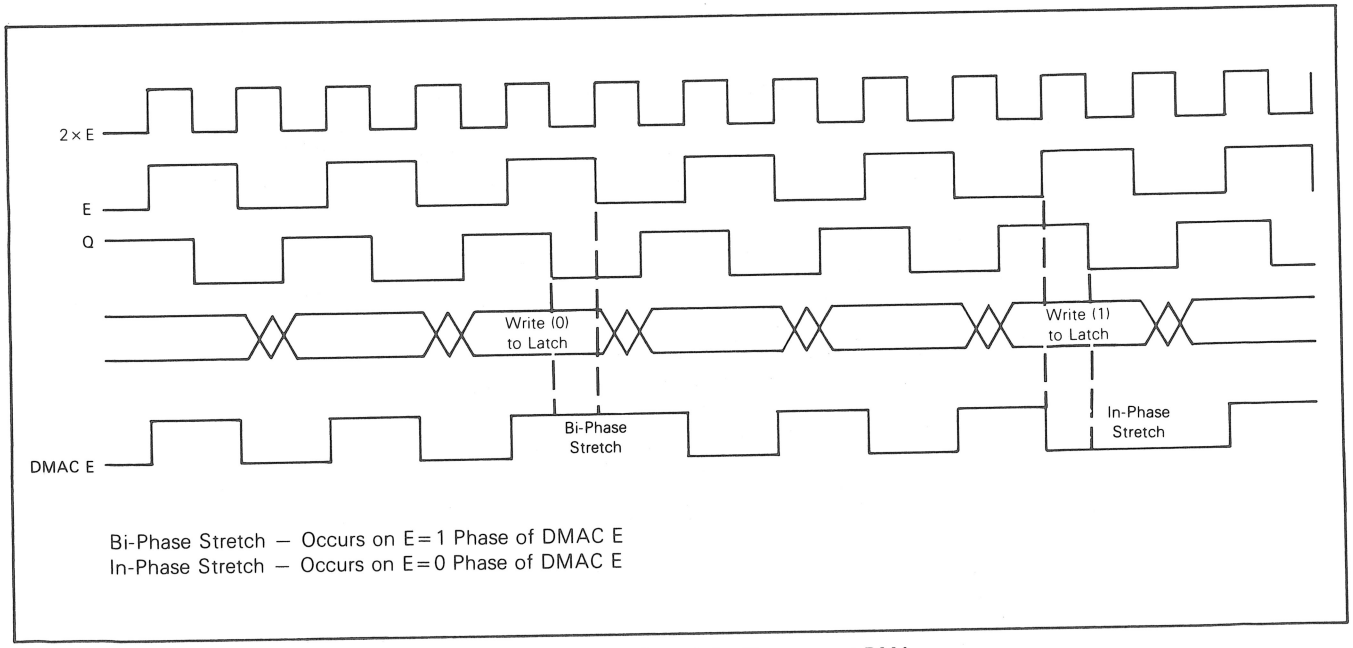
2 × E
E
Q
Write (0) to Latch
Write (1) to Latch
Bi-Phase Stretch
In-Phase Stretch
DMAC E
Bi-Phase Stretch — Occurs on E = 1 Phase of DMAC E
In-Phase Stretch — Occurs on E = 0 Phase of DMAC E

FIGURE 4 — Clock Stretch for Transparent DMA

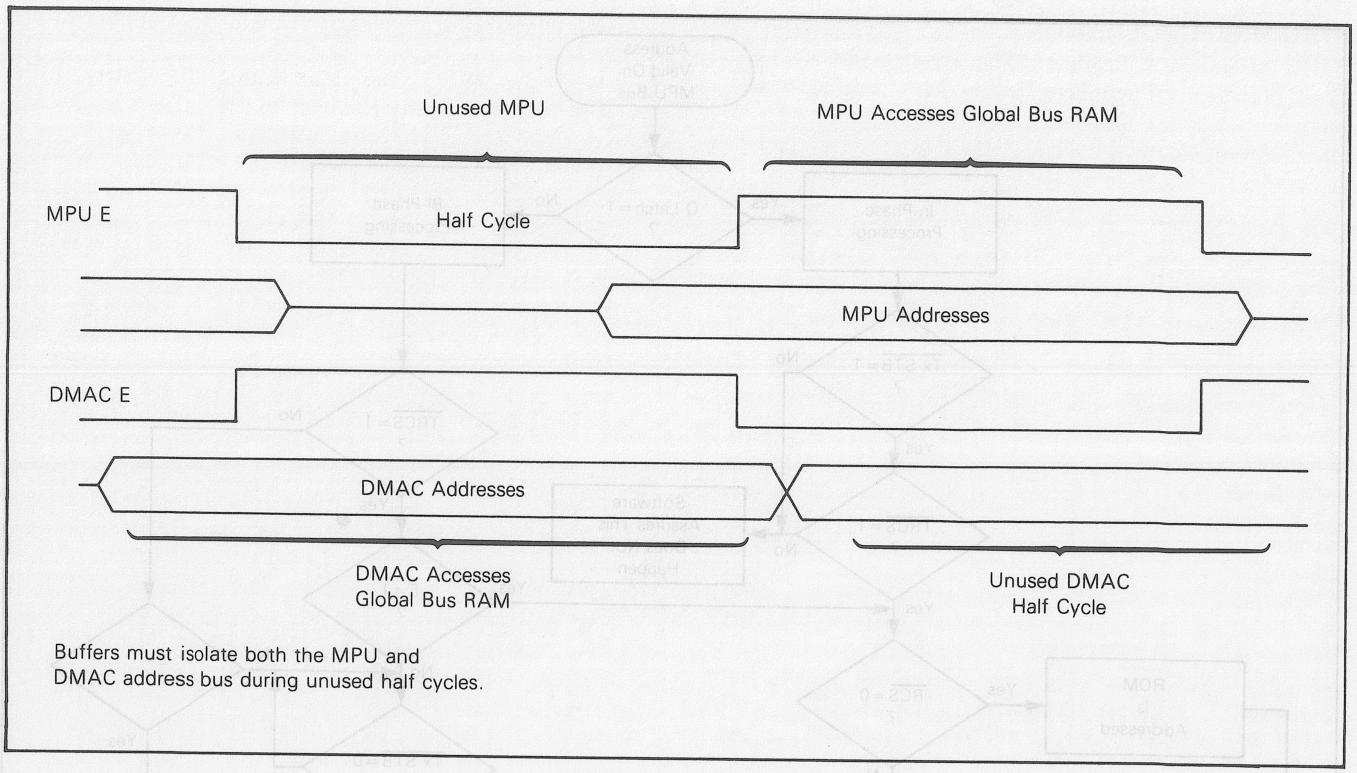


**FIGURE 5 — Bi-phase Timing**

| SIGNAL LINE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| MPU E | $\overline{DMACS}$ | $\overline{TRCS}$ | $\overline{Tx\ STB}$ | MPU $\overline{G}$ | DMAC $\overline{G}$ ADDRESS | SCSI $\overline{G}$ & DATA | DMAC $\overline{G}$ DATA | NOTES |
| 0 | 0 | 0 | 0 | X | X | X | X | A |
| 0 | 0 | 0 | 1 | X | X | X | X | B |
| 0 | 0 | 1 | 0 | X | X | X | X | A |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | – |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | – |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | – |
| 0 | 1 | 1 | 0 | X | X | X | X | C |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | – |
| 1 | 0 | 0 | 0 | X | X | X | X | B |
| 1 | 0 | 0 | 1 | X | X | X | X | B |
| 1 | 0 | 1 | 0 | X | X | X | X | A |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | – |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | – |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | – |
| 1 | 1 | 1 | 0 | X | X | X | X | C |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | – |

NOTES: (A) DMAC cannot be selected while transfer is in progress.

(B) $\overline{DMACS}$ and $\overline{TRCS}$ are not asserted simultaneously.

(C) Software assures this never happens.

(D) Turn MPU buffers on when MPU initializes DMAC ($\overline{DMACS}=0$) or when MPU reads and writes the table RAM ($\overline{TRCS}=0$ and E= 1).

(E) Turn DMAC address buffers on when MPU initializes the DMAC ($\overline{DMACS}=0$) or when $\overline{Tx\ STB}$ is active and DMAC E is high ($\overline{Tx\ STB}=0$ and MPU E=0).

(F) Turn DMAC data buffers on when MPU initializes DMAC ($\overline{DMACS}=0$).

(H) Turn SCSI data buffers on when $\overline{Tx\ STB}$ is active and DMAC E is high ($\overline{Tx\ STB}=0$ and E=0).

**FIGURE 6 — Truth Table for Control of Bus Buffers**

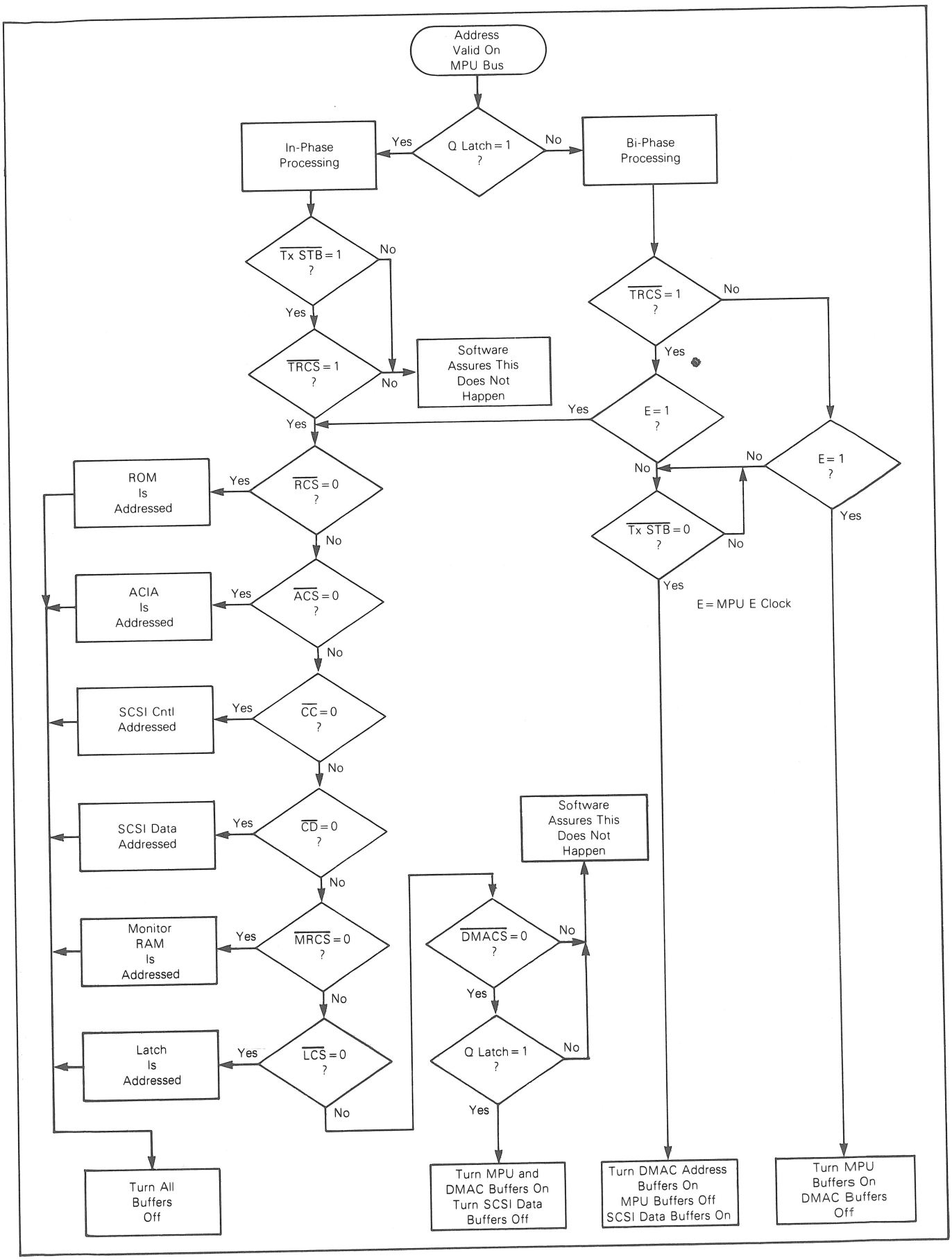


FIGURE 7 — Flowchart for Bus Buffer Control

The MPU data bus is also buffered to and from the SCSI. These buffers are turned on when the MPU is reading or writing a sector or transferring a command. With this design implementation, both the MPU and DMAC can read a sector. However, in this application note, only the MPU can transfer the actual read command to the disk storage subsystem (via the SCSI).

To perform transparent DMA, it is necessary to stretch the E clock (DMAC E) applied to the DMAC to ensure that it is 180-degrees out-of-phase with the MPU E clock. In addition, there must be no glitches on the stretched E clock. Thus, a hardware design utilizing a multiplexer is automatically eliminated. The multiplexer switches its output between two selected inputs and a glitch could occur at switching time.

The stretched E clock in this application note is implemented using two D-type flip-flops (an SN74LS74) in a programmable latch configuration. One flip-flop is configured as a programmable 1-bit latch with its output selecting either the MPU clock or the complement of the MPU clock via a gating function.

Selection of the clock is performed by an exclusive OR gate which provides an output which is either in-phase or out-of-phase with the MPU E clock. The exclusive OR output is applied to the second flip-flop which is clocked by the $2 \times E$ clock (twice the E clock frequency) to eliminate the possibility of glitches in the DMAC E clock. The flip-flop which provides the DMAC E clock acts as a programmable latch and is written during the fourth quadrant ($Q = 0$, $E = 1$) of the MPU (see Figure 4).

After reset, the DMAC E clock is always in-phase with the MPU E clock. When DMA is required, the DMAC E clock is then stretched into the 180-degree out-of-phase scheme during the fourth quadrant of the MPU E and Q clocks. Timing is such that the out-of-phase scheme only occurs when the DMAC E clock is high and is returned to the in-phase state only when the DMAC E clock is low. This is illustrated in the timing diagram of Figure 4. Note that the change from in-phase to out-of-phase, or out-of-phase to in-phase can only occur during a write to the latch ($\overline{W}$ is low, $\overline{CSCS}$ is low, and $\overline{Q}$ clock is low).

The $\overline{Q}$ output of the first half of the programmable latch is used to select the direction of the bidirectional address buffer on the DMAC address bus. When the MPU E clock is in-phase with the DMAC E clock, the output of the buffers is applied to the DMAC (B to A on the SN74LS245s) and the low order address lines of the DMAC provide the register address of the DMAC for register reads and writes. When the MPU E clock is out-of-phase with the DMAC E clock, the buffers reverse direction (B to A on the SN74LS245s) and the DMAC address lines are output to the MCM6116 and CMOS static RAM on the global bus.

The control signals for the MCM6116 CMOS static RAM, which are wired to the MPU local bus, are derived from the $R/\overline{W}$ and MPU E clock signals in addition to the address decoded by the SN74LS138 (1-of-8 decoder). Care should be taken in qualifying read and writes to the RAM with the MPU E clock. A general $\overline{RD}$ and $\overline{W}$ should be derived to interface with $\overline{OE}$ and $\overline{W}$, respectively, on the RAM to avoid extraneous reads and writes, while $\overline{CS}$ should be generated by the address decoder (output of SN74LS138).

The control of the MCM6116 CMOS static RAM located on the global bus is shared by the MPU and the DMAC. The $\overline{CS}$ is generated from either the SN74LS138 on the MPU local address bus or by $\overline{Tx\ STB}$ of the DMAC. Since the DMAC is only addressing the RAM, address decoding on the global bus is eliminated.

The MPU and DMAC buffers turn on and off alternately while DMA is in progress. Therefore, it is necessary to delay a write to the RAM past the rising edge of the E clock of both the MPU and DMAC. This delay is necessary because of the propagation delay through the address buffer. Addresses must not be changing when a write occurs to the RAM since this will cause an extraneous write to memory.

In this application, the write is delayed until the fourth quadrant of the MC68B09E for the MPU and the second quadrant of the MC68B09E for the DMAC. An exclusive OR gate with inputs from the MC68B09E E and Q clocks is used to accomplish this delay since its output will be high only in the second and fourth quadrants. The output enable ($\overline{G}$) signal for the RAM is generated using the $R/\overline{W}$ line of the global bus.

In this bi-phase configuration, $\overline{G}$ of the shared RAM cannot be qualified with the MPU E clock. Since the DMAC E clock is out of phase with the MPU E clock, this would prohibit the DMAC from accessing memory. Therefore, the output buffers are enabled at all times on the RAM except when global $R/\overline{W} = 0$ occurs (first and third quadrants) at which time they are disabled.

The control circuitry which interfaces the DMAC with the SCSI bus is configured using four D-type flip-flops (SN74LS74s). Two of the flip-flops perform the asynchronous handshake required by the SCSI, one flip-flop generates a DMA GRANT (DGRANT) for the DMAC (the MPU is not halted as in normal operation therefore it does not generate the DGRANT signal), and one flip-flop generates $\overline{IRQ}$ for the MPU.

Since DGRANT is not generated by the MPU in the normal process of decoding HALT acknowledge during the dead cycle of the DMAC, the recommended circuitry described in the MC6844 data sheet for the generation of $\overline{IRQ}$ can not be used in this application. The circuit used to demultiplex the DEND/$\overline{IRQ}$ input to the DMAC allows DEND/$\overline{IRQ}$ to clock a flip-flop, which generates the $\overline{IRQ}$ signal to the MPU, and the interrupt acknowledge (IACK) on the MPU negates $\overline{IRQ}$.

Since the $\overline{IRQ}$ line of the MPU is only driven by one source in this application note, the output of the flip-flop can be connected directly to the MPU without being buffered by an open collector gate. If more than one device were able to interrupt the MPU on its IRQ line, then an open collector gate would be required.

The asynchronous $\overline{REQ}$ signal of the SCSI bus generates the Tx RQ. Refer to the timing diagram shown in Figure 8. When $\overline{REQ}$ becomes active, it enables the flip-flop which asserts Tx RQ. An unstable state can occur at the output of a flip-flop if the D input is asynchronous and the setup time with respect to the rising clock edge is not met.

In this design, the flip-flop which generates Tx RQ is held in a clear state until $\overline{REQ}$ becomes active. Once $\overline{REQ}$ becomes active, the flip-flop is removed from the clear state. Thus, $\overline{REQ}$ will be setup properly when the flip-flop is clocked on the rising edge of the next DMAC E clock. The Tx RQ will be setup properly for DMA timing when it is sampled on the falling edge of the DMAC E clock.

The initial sample of Tx RQ by the DMAC occurs on the rising edge of the DMAC E clock while all subsequent samples occur on the falling edge of this clock. Assertion of DRQ2 by the DMAC is delayed by one DMAC E clock

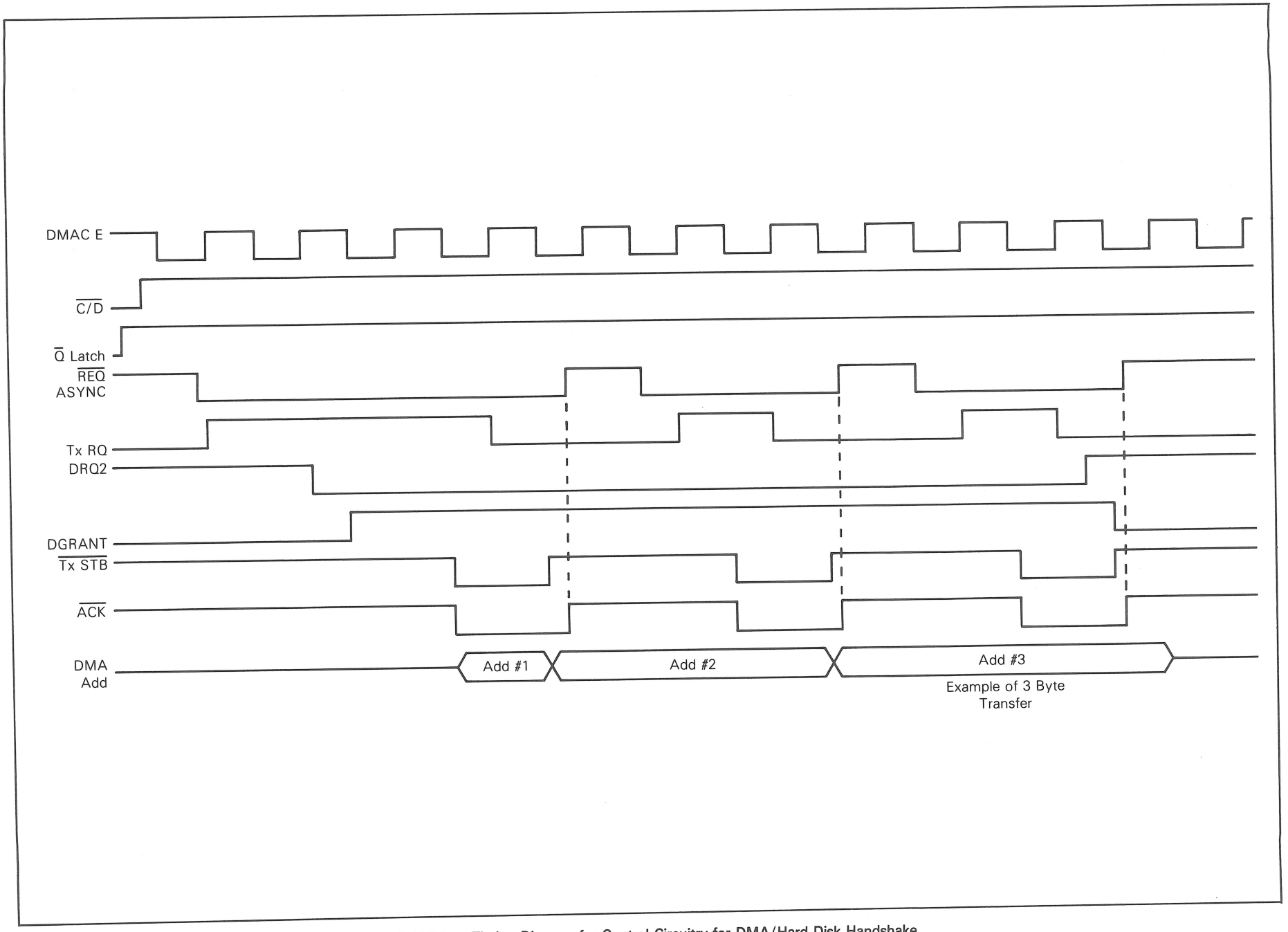


FIGURE 8 — Timing Diagram for Control Circuitry for DMA/Hard Disk Handshake

because Tx RQ is sampled on the rising edge of the DMAC E clock and Tx RQ will not meet the proper setup time.

The Tx RQ is deasserted when $\overline{\text{Tx STB}}$ is asserted and the DMAC E clock is high, thus clearing the flip-flop which generates Tx RQ. The Tx RQ must be deasserted before the falling edge of the DMAC E clock; otherwise, the address will be incremented on the DMAC by the next falling edge of the DMAC E clock and synchronization of data from the SCSI bus will be lost. A byte of data must be ready for transfer each time the address on the DMAC address bus increments.

The Tx RQ will then remain deasserted until the next falling edge of $\overline{\text{REQ}}$ is followed by a rising edge of the DMAC E clock. At this time, the Tx RQ line will again be asserted to provide synchronization with the data from the SCSI bus. The $\overline{\text{ACK}}$ signal generated by this circuit uses $\overline{\text{Tx STB}}$ from the DMAC and $\overline{\text{REQ}}$ from the SCSI bus. The $\overline{\text{Tx STB}}$ cannot be used as the $\overline{\text{ACK}}$ signal since it will only be asserted for one cycle at a time.

The SCSI specification prohibits the deassertion of $\overline{\text{ACK}}$ before REQ is deasserted. This is accomplished using a SR type cross-coupled NAND gate flip-flop, allowing $\overline{\text{Tx STB}}$ to assert $\overline{\text{ACK}}$ while $\overline{\text{REQ}}$ controls the deassertion of $\overline{\text{ACK}}$. All preset and clear inputs to the flip-flops which use the $\overline{\text{RESET}}$ signal are buffered by a Schmitt trigger because of the slow rise time on the $\overline{\text{RESET}}$ line.

## SYSTEM EXPLANATION AND THEORY

The system can be used to increase bus utilization by providing transparent accesses to RAM. The design using a hard disk storage subsystem does not implement a truly transparent system since the MPU must first initiate a transfer by sending a command to the disk storage subsystem. When the disk storage subsystem has transferred all the data, it then interrupts the MPU which must then read the status of the disk storage subsystem. the SCSI which must then read the status of the disk storage subsystem. The SCSI specification provides for a status message following all data transfers which the MPU must read from the disk storage subsystem. Refer to the SCSI specifications for specific interface timing.

The primary purpose of this application note is to show that the M6809 Family timing will allow two memory accesses during one MPU E clock cycle. The disk storage subsystem is used since its asynchronous signals will interface easily with the synchronous signals of the DMAC. Also, the type of data desired for this application can be generated while the MPU is processing some other routine algorithm.

The system in this application note cannot be interfaced with a peripheral I/O controller from the M6800 Family, e.g., MC6854, etc., since the E clock phase of the peripheral controller would require continuous changing with respect to the MPU. However, any asynchronous device such as an A/D converter can be interfaced with this system using the same control signals since all this system requires is a device which can generate a request for data and then accept an acknowledge of data transfer.

## SOFTWARE DESCRIPTION

A simple routine is provided in this application note which demonstrates the functionality of using the transparent DMA system discussed. The algorithm simply reads a sector of data from the disk storage subsystem. The DMAC then interrupts the MPU which performs the complement of the data previously written into RAM.

At the same time that the MPU is performing the operation on the previously generated data, the DMAC is generating a new table of data. When the algorithm is completed, a jump occurs back to the monitor which allows examination of memory to determine if the entire operation was accomplished.

The software begins with the initialization of the disk storage subsystem as shown in Figure 9. This is followed by initialization of the DMAC registers as shown in Figure 10. The address and byte count registers determine the starting address for DMA and the number of bytes to be transferred, respectively.

The channel control register serves multiple functions. Mode control bits in the channel control register determine the mode in which the DMAC will operate. For this application, the halt-burst mode is used. The channel control register also chooses whether the DMAC will be reading or writing memory. The direction of the address pointer is selected to allow the addresses to either increment or decrement.

The priority control register simply enables one or more of the several channels on the DMAC. The interrupt control register enables an interrupt on the channel which is acquiring DMA so that it may interrupt the processor at the end of the DMA cycle. In this application note, only channel 0 is enabled and allowed to interrupt the MPU. The data chain register selects the two or four channel mode. Since only one channel is being used, the two channel mode was selected. The software then performs the clock stretch to the DMAC so that DMAC E clock is 180-degrees out-of-phase with the MPU E clock.

When the DMAC is properly setup to perform DMA (initialization has occurred), the MPU must then transfer a command to the disk storage subsystem. In this application, the command is a read since a sector will be read from the disk storage subsystem. The MPU will first write a sector to the disk storage subsystem so that data will be available to compare when the DMA process is over.

When the command is sent to the disk storage subsystem, two operations are being performed simultaneously and the MPU will run under normal operation unaffected by the DMAC operation. Concurrently, the DMAC, in conjunction with the disk operating subsystem, performs data transfer unaffected by the MPU. The DMAC asserts IRQ upon completion of a data transfer.

The MPU interrupt service routine first receives the status byte from the disk operating system and the DMAC E clock is then stretched back in-phase with the MPU E clock. This allows the reading and writing of the DMAC registers. The MPU then clears the interrupt by reading the channel control register, DMAC initialization for the new table begins, and the previous sequence is repeated as long as DMA is required.

## CONCLUSION

The ability of a DMAC to perform transparent RAM accesses increases bus utilization. By allowing a DMAC to interleave its clock cycle with that of the MPU in the biphase

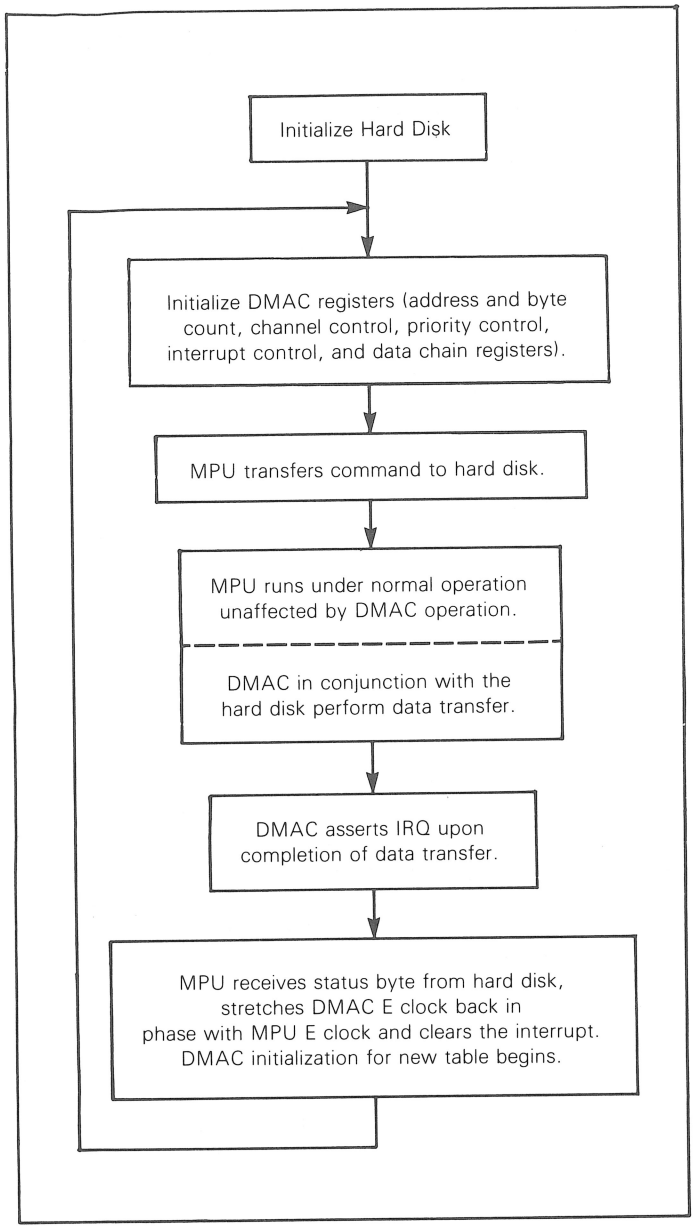
Initialize Hard Disk
Initialize DMAC registers (address and byte count, channel control, priority control, interrupt control, and data chain registers).
MPU transfers command to hard disk.
MPU runs under normal operation unaffected by DMAC operation.
DMAC in conjunction with the hard disk perform data transfer.
DMAC asserts IRQ upon completion of data transfer.
MPU receives status byte from hard disk, stretches DMAC E clock back in phase with MPU E clock and clears the interrupt. DMAC initialization for new table begins.

FIGURE 9 — Software Flowchart

configuration described in this application note, the number of applications for the already powerful M6809 Family is increased.

For example, with a 2-MByte/second transfer rate, applications involving fast Fourier transforms on data acquired by an A/D converter could be used to build a spectrum analyzer. This biphase principle could also be used in video applications involving bit mapped graphics. Numerous frames of data could be stored on hard disk. The data from the hard disk could then be transferred to display RAM transparent to the MPU.

These are only a few of the applications possible with this bi-phase concept. This application note thus provides the designer with the fundamental concepts necessary to design a MPU system with DMA accesses which are transparent to the MPU.

| Register | Channel | Address (Hex) |
|---|---|---|
| Address High | 0 | 0 |
| Address Low | 0 | 1 |
| Byte Count High | 0 | 2 |
| Byte Count Low | 0 | 3 |
| Address High | 1 | 4 |
| Address Low | 1 | 5 |
| Byte Count High | 1 | 6 |
| Byte Count Low | 1 | 7 |
| Address High | 2 | 8 |
| Address Low | 2 | 9 |
| Byte Count High | 2 | A |
| Byte Count Low | 2 | B |
| Address High | 3 | C |
| Address Low | 3 | D |
| Byte Count High | 3 | E |
| Byte Count Low | 3 | F |

FIGURE 10A — Address and Byte Count Registers

| Register | Address (Hex) | Register Content | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Channel Control | 1x* | DMA End Flag (DEND) | Busy/$\overline{\text{Ready}}$ Flag | Not Used | Not Used | $\overline{\text{Address}}$ Up/Down | MCA | MCB | Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$) |
| Priority Control | 14 | Rotate Control | Not Used | Not Used | Not Used | Request Enable #3 (RE3) | Request Enable #2 (RE2) | Request Enable #1 (RE1) | Request Enable #0 (RE0) |
| Interrupt Control | 15 | DEND IRQ Flag | Not Used | Not Used | Not Used | DEND IRQ Enable #3 (DIE3) | DEND IRQ Enable #2 (DIE2) | DEND IRQ Enable #1 (DIE1) | DEND IRQ Enable #0 (DIE0) |
| Data Chain | 16 | Not Used | Not Used | Not Used | Not Used | Two/Four Channel Select (2/4) | Data Chain Channel Select B | Data Chain Channel Select A | Data Chain Enable |

*The x represents the binary equivalent of the channel desired.

FIGURE 10B — DMAC Control Registers

```
---


PAGE  001  TDMA6   .SA:1

00001              3800     A CNTL   EQU   $3800
00002A FFF8                 ORG   $FFF8
00003A FFF8        F044     A        FDB   ISR
00004              4000     A DMAT   EQU   $4000
00005              3400     A LATCH  EQU   $3400
00006              0000     A ADH    EQU   0
00007              0002     A BCH    EQU   2
00008              0010     A CHC    EQU   16
00009              0014     A PRC    EQU   20
00010              0015     A ICC    EQU   21
00011              0016     A DCC    EQU   22
00012              8000     A DMAC   EQU   $8000
00013              3C00     A DATA   EQU   $3C00
00014                       *
00015A 6700                 ORG   $6700
00016                       *
00017A 6700        000F     A COMND  RMB   15
00018A 670F        0001     A STATUS RMB   1
00019A 6710        0002     A SAVEX  RMB   2
00020A 6712        0002     A MAXD   RMB   2
00021A 6714        0002     A SAVEX1 RMB   2
00022                       *
00023A 6400                 ORG   $6400
00024                       *
00025A 6400        0100     A BYTEST RMB   256    HERE FOR THE REAL STUFF
00026                       *
00027                       *
00028                       *
00029A 6500                 ORG   $6500
00030                       *
00031A 6500        0100     A RDTEST RMB   256    HERE FOR THE READ STUFF
00032                       *
00033                       *  BEGINNING OF PROGRAM DATA
00034                       *
00035                       *
00036A F000                 ORG   $F000
00037                       *
00038A F000 4F              INIT  CLRA
00039A F001 B7   3C00       A     STA   DATA     STORE A 00
00040A F004 B7   3800       A     STA   CNTL     HERE TOO
00041                       *
00042                       *  RESET CONTROLLER
00043                       *
00044A F007 86   02         A     LDA   #$02     ASSERT RESET
00045A F009 B7   3800       A     STA   CNTL
00046A F00C 4F              A     CLRA
00047A F00D B7   3800       A     STA   CNTL     DE ASSERT IT
00048A F010 39              A     RTS
00049                       *
00050                       *       INITIALIZE DMAC FOR 256 BYTE TRANSFER
00051                       *       ENTER WITH U POINTING TO DMAC
00052
00053                       *
00054A F011 30   89 4000 A DMA   LEAX  DMAT,X
00055A F015 1F   10         A     TFR   X,D
00056A F017 ED   C4         A     STD   ADH,U
00057A F019 CC   0100       A     LDD   #$0100   SET UP 256 BYTES OF DMA TRANSFER
00058A F01C ED   42         A     STD   BCH,U    SEND # OF BYTES TO DMAC
```

```
---


PAGE  002  TDMA6   .SA:1

00059A F01E 86   02      A        LDA    #$02     SET UP DMAC FOR HALT BURST MODE
00060A F020 A7   C8 10   A        STA    CHC,U    R/W IS WRITE AND ADDRESS WILL INC
00061A F023 86   01      A        LDA    #$01     ENABLE CHANNEL 0 OF DMA
00062A F025 A7   C8 14   A        STA    PRC,U    SET FIXED PRIORITY IF ITS WORTH IT
00063A F028 A7   C8 15   A        STA    ICC,U    ENABLE INTERRUPT ON CHANNEL 0
00064A F02B 4F                    CLRA
00065A F02C A7   C8 16   A        STA    DCC,U    SELECT 2 CHANNEL MODE AND DISABLE
00066A F02F B7   3400    A        STA    LATCH
00067A F032 39                    RTS
00068                     **********************************************
00069                     **********************************************
00070                     ***THIS ROUTINE COMPLEMENTS AN OLD
00071                     ***TABLE WHILE THE NEW TABLE IS FILLING
00072A F033 A6   89 4000 A COMP   LDA    DMAT,X   LOAD TABLE VALUE
00073A F037 43                    COMA
00074A F038 A7   89 4000 A        STA    DMAT,X   STORE BACK TO SAME ADD
00075A F03C 30   01               INX
00076A F03E 8C   0100    A        CMPX   #$100    256 BYTES YET
00077A F041 26   F0   F033        BNE    COMP
00078A F043 39                    RTS
00079                     *
00080                     *
00081                     *** THIS IS THE INTERRUPT SERVICE ROUTINE
00082                     *** ITS FUNCTION IS TO PLACE MPU AND DMA
00083                     *** CLOCKS IN PHASE, CLEAR DMA INT AND
00084                     *** SET UP THE NEW DMAC TO ADD
00085                     **********************************************
00086                     **********************************************
00087A F044 BD   F0C5    A ISR    JSR    STAMSG   GET STATUS BYTE
00088A F047 86   FF      A        LDA    #$FF     CLOCKS IN PHASE
00089A F049 B7   3400    A        STA    LATCH
00090A F04C A6   C8 10   A        LDA    CHC,U    CLEAR INT
00091A F04F 8E   0100    A        LDX    #$100    SET UP 2ND TABLE ADD
00092A F052 CE   8000    A        LDU    #DMAC
00093A F055 BD   F011    A        JSR    DMA
00094A F058 3B                    RTI
00095                     **********************************************
00096                     **********************************************
00097                     *
00098                     *
00099                     *  WAIT TILL NOT BUSY
00100                     *
00101                     *
00102A F059 B6   3800    A WNBUSY LDA    CNTL     GET STATUS
00103A F05C 46                    RORA
00104A F05D 25   FA   F059        BCS    WNBUSY   STILL BUSY
00105A F05F 39                    RTS
00106                     *
00107                     *
00108                     *  WAIT FOR BUSY
00109                     *
00110                     *
00111A F060 B6   3800    A WBUSY  LDA    CNTL
00112A F063 46                    RORA
00113A F064 24   FA   F060        BCC    WBUSY    WAIT TILL BUSY
00114A F066 39                    RTS             THEN RETURN
```

```
---


PAGE  003  TDMA6    .SA:1

00116                          *
00117                          *
00118                          *  WAIT FOR REQUEST
00119                          *
00120                          *
00121A F067 34    02           WREQ   PSHA
00122A F069 B6    3800     A   WREQ1  LDA    CNTL
00123A F06C 46                       RORA
00124A F06D 46                       RORA
00125A F06E 46                       RORA           GET INTO CARRY
00126A F06F 46                       RORA
00127A F070 24    F7   F069         BCC    WREQ1   WAIT TILL REQUEST
00128A F072 35    02                 PULA
00129A F074 39                       RTS
00130                          *
00131                          *
00132                          *  WAIT FOR NOT REQUEST
00133                          *
00134                          *
00135A F075 34    02           WNREQ  PSHA
00136A F077 B6    3800     A   WNREQ1 LDA    CNTL
00137A F07A 46                       RORA
00138A F07B 46                       RORA
00139A F07C 46                       RORA
00140A F07D 46                       RORA
00141A F07E 25    F7   F077         BCS    WNREQ1  WAIT TILL NOT REQUEST
00142A F080 35    02                 PULA
00143A F082 39                       RTS
00144                          *
00145                          *
00146                          *  SELECT
00147                          *
00148                          *
00149A F083 BD    F059     A SELECT JSR    WNBUSY  WAIT TILL NOT BUSY
00150A F086 86    01       A        LDA    #$01
00151A F088 B7    3C00     A        STA    DATA    ASSERT CONTROLLER ID
00152A F08B 86    04       A        LDA    #$04
00153A F08D B7    3800     A        STA    CNTL    ASSERT SEL
00154                          *
00155A F090 BD    F060     A        JSR    WBUSY   WAIT TILL BUSY
00156A F093 4F                     CLRA
00157A F094 B7    3800     A        STA    CNTL    DE ASSERT SEL
00158A F097 B7    3C00     A        STA    DATA    DE ASSERT CONTROLLER ID
00159A F09A 39                     RTS            NOW READY FOR OTHER STUFF
```

```
---


PAGE  004  TDMA6    .SA:1

00161                          *
00162                          *
00163                          *  TXFCMD  TRANSFER COMMAND
00164                          *
00165                          *  ACCB = # BYTES TO SEND
00166                          *  X REG = LOCATION OF THOSE TO SEND
00167                          *
00168                          *
00169A F09B BF    6710     A TXFCMD STX     SAVEX
00170A F09E 3A                    ABX              ADD THE OFFSET TO POINTER
00171A F09F BF    6712     A       STX     MAXD     PUT IN MAX POINTER
00172A F0A2 BE    6710     A       LDX     SAVEX    RESTORE BEGINNING
00173                          *
00174A F0A5 BD    F0EB     A TXLOOP JSR     CHKCD
00175A F0A8 24    FB    F0A5       BCC     TXLOOP   WAIT FOR CD TO DEASSERT
00176A F0AA BD    F107     A TXLP1  JSR     CHKIO    SEE IF I/O ASSERTED
00177A F0AD 24    FB    F0AA       BCC     TXLP1    ,,, THEN WAIT TILL HE DOES
00178A F0AF BD    F067     A       JSR     WREQ
00179A F0B2 BD    F067     A       JSR     WREQ
00180A F0B5 A6    84       A       LDA     0,X      GET 1ST BYTE
00181A F0B7 B7    3C00     A       STA     DATA
00182A F0BA BD    F0F6     A       JSR     ACKNOW
00183A F0BD 30    01               INX
00184A F0BF BC    6712     A       CPX     MAXD
00185A F0C2 26    E1    F0A5       BNE     TXLOOP   NOT DONE YE'T
00186A F0C4 39                     RTS              COMMAND COMPLETE
00187                          *
```

```
---


PAGE  005  TDMA6    .SA:1

00189                          *
00190                          *  GET STATUS MESSAGE
00191                          *
00192A F0C5 BD    F0EB     A STAMSG JSR     CHKCD
00193A F0C8 24    FB    F0C5       BCC     STAMSG   WAIT FOR CD TO ASSERT
00194A F0CA BD    F067     A       JSR     WREQ     WAIT FOR REQUEST
00195A F0CD B6    3C00     A       LDA     DATA     READ STATUS BYTE
00196A F0D0 B7    670F     A       STA     STATUS   SAVE IT AWAY
00197                          *
00198A F0D3 BD    F0F6     A       JSR     ACKNOW   GO ASSERT & DEASSERT
00199                          *
00200A F0D6 12                     NOP              ?????
00201                          *
00202                          *
00203A F0D7 B6    3800     A STLOOP LDA     CNTL
00204A F0DA 46                     RORA
00205A F0DB 46                     RORA
00206A F0DC 24    F9    F0D7       BCC     STLOOP   MSG ASSERTED?
00207A F0DE BD    F067     A       JSR     WREQ     YEP  WAIT FOR REQUEST
00208A F0E1 B6    3C00     A       LDA     DATA
00209A F0E4 B7    6710     A       STA     STATUS+1
00210                          *
00211A F0E7 BD    F0F6     A       JSR     ACKNOW
00212                          *
00213A F0EA 39                     RTS              NOW READY FOR ANOTHER COMMAND
00214                          *
00215                          *
```

```
---


PAGE  006  TDMA6   .SA:1

00217                          *
00218A F0EB 34   02      CHKCD  PSHA
00219A F0ED B6   3800    A      LDA    CNTL
00220A F0F0 46                  RORA                IF CD = 1 THEN COMMAND
00221A F0F1 46                  RORA                IF CD = 0 THEN DATA
00222A F0F2 46                  RORA
00223A F0F3 35   02             PULA
00224A F0F5 39                  RTS                 RETURN WITH CARRY
00225                          *
00226                          *
00227                          *
00228A F0F6 34   02      ACKNOW PSHA
00229A F0F8 86   01      A      LDA    #$01
00230A F0FA B7   3800    A      STA    CNTL
00231A F0FD BD   F075    A      JSR    WNREQ
00232A F100 4F                  CLRA
00233A F101 B7   3800    A      STA    CNTL
00234A F104 35   02             PULA
00235A F106 39                  RTS
00236                          *
00237                          *  CHECK IO LINE
00238                          *
00239A F107 B6   3800    A CHKIO  LDA    CNTL        GET STATUS
00240A F10A 49                  ROLA
00241A F10B 49                  ROLA
00242A F10C 49                  ROLA
00243A F10D 49                  ROLA
00244A F10E 39                  RTS
00245                          *
```

```
---


PAGE  007  TDMA6   .SA:1

00247                          *
00248                          *
00249                          *
00250                          *  FORMAT DRIVE COMMAND
00251                          *
00252                          *
00253A F10F BD   F083    A FORMAT JSR    SELECT    SELECT CONTROLLER
00254A F112 8E   0400    A        LDX    #$0400
00255A F115 BF   6700    A        STX    COMND
00256                          *
00257A F118 8E   0000    A        LDX    #$0000
00258A F11B BF   6702    A        STX    COMND+2
00259                          *
00260A F11E 8E   0000    A        LDX    #$0000
00261A F121 BF   6704    A        STX    COMND+4   NOW ALL SIX BYTES SETUP
00262                          *
00263A F124 C6   06      A        LDB    #$06
00264                          *
00265                          *
00266A F126 8E   6700    A        LDX    #COMND
00267A F129 BD   F09B    A        JSR    TXFCMD
00268A F12C BD   F0C5    A        JSR    STAMSG
00269A F12F 39                    RTS
00270                          *
```

```
---


PAGE  008  TDMA6    .SA:1

00272                          *
00273                          *  WRITE A SINGLE 256 BYTE SECTOR
00274                          *  ENTER WITH X = TRACK ADDRESS
00275                          *
00276                          *
00277A F130 BD    F083     A WTSEC  JSR    SELECT   GO SELECT CONTROLLER
00278                          *
00279A F133 108E 0A00      A        LDY    #$0A00
00280A F137 10BF 6700      A        STY    COMND
00281                          *
00282A F13B BF    6702     A        STX    COMND+2
00283                          *
00284A F13E 108E 0100      A        LDY    #$0100
00285A F142 10BF 6704      A        STY    COMND+4
00286A F146 C6    06       A        LDB    #$06     NUMBER OF BYTES TO DO
00287                          *
00288A F148 8E    6700     A        LDX    #COMND   LOCATION OF THEM
00289A F14B BD    F09B     A        JSR    TXFCMD
00290                          *
00291A F14E 8E    6400     A        LDX    #BYTEST  WHERE TO WRITE FROM
00292                          *
00293A F151 BD    F0EB     A WRDATA JSR    CHKCD    CD ASSERTED?
00294A F154 25    FB    F151        BCS    WRDATA   WAIT TILL DEASSERTS
00295                          *
00296A F156 BD    F067     A WDLOOP JSR    WREQ
00297A F159 A6    84       A        LDA    0,X
00298A F15B B7    3C00     A        STA    DATA     SEND IT OUT
00299A F15E BD    F0F6     A        JSR    ACKNOW   ACKNOWLEDGE IT
00300A F161 30    01                INX
00301A F163 8C    6500     A        CPX    #BYTEST+256 MAXD OUT?
00302A F166 26    EE    F156        BNE    WDLOOP
00303                          *
00304A F168 BD    F0C5     A        JSR    STAMSG   GO GET STATUS
00305                          *
00306A F16B 39                     RTS
```

```
---


PAGE  009  TDMA6    .SA:1

00308                          *
00309                          *  READ A SINGLE 256 BYTE SECTOR
00310                          *
00311                          *  ENTER WITH X = LOGICAL ADDRESS
00312                          *
00313A F16C BD   F083     A RDSEC  JSR    SELECT   GO SELECT CONTROLLER
00314                          *
00315A F16F 108E 0800     A        LDY    #$0800
00316A F173 10BF 6700     A        STY    COMND
00317                          *
00318A F177 BF   6702     A        STX    COMND+2  HERES WHERE THE TRAC ADDR IS
00319                          *
00320A F17A 108E 01C0     A        LDY    #$01C0
00321A F17E 10BF 6704     A        STY    COMND+4
00322A F182 C6   06       A        LDB    #$06     NUMBER OF BYTES OF COMMAND
00323                          *
00324A F184 8E   6700     A        LDX    #COMND   WHERE THEY ARE
00325A F187 BD   F09B     A        JSR    TXFCMD   SEND COMMAND
00326A F18A 8E   6500     A        LDX    #RDTEST  THE DATA TRANSFER AREA
00327                          *
00328A F18D BD   F0EB     A REDATA JSR    CHKCD    WAIT FOR CD TO DEASSERT
00329A F190 25   FB   F18D        BCS    REDATA
00330                          *
00331                          *
00332A F192 BD   F067     A RDLOOP JSR    WREQ
00333A F195 B6   3C00     A        LDA    DATA     GET DATA BYTE
00334A F198 A7   84       A        STA    0,X      PUT IT IN MEMORY
00335A F19A 30   01               INX
00336A F19C BD   F0F6     A        JSR    ACKNOW   GO ACKNOWLEDGE IT
00337A F19F 8C   6600     A        CPX    #RDTEST+256 END YET?
00338A F1A2 26   EE   F192        BNE    RDLOOP   GO GET ANOTHER
00339                          *
00340A F1A4 BD   F0C5     A        JSR    STAMSG   GET STATUS
00341A F1A7 39                    RTS
00342                          **********THIS IS THE DMA READ ROUTINE**********
00343                          *********                          **********
00344A F1A8 BD   F083     A DREAD  JSR    SELECT
00345A F1AB 108E 0800     A        LDY    #$0800
00346A F1AF 10BF 6700     A        STY    COMND
00347A F1B3 BF   6702     A        STX    COMND+2
00348A F1B6 108E 01C0     A        LDY    #$01C0
00349A F1BA 10BF 6704     A        STY    COMND+4
00350A F1BE C6   06       A        LDB    #$06
00351A F1C0 8E   6700     A        LDX    #COMND
00352A F1C3 BD   F09B     A        JSR    TXFCMD
00353A F1C6 39                    RTS
00354                          ************************************************
00355                          *
```

```
---


PAGE  010  TDMA6    .SA:1

00357                          *  THIS ROUTINE STORES CONSECUTIVE
00358                          *  NUMBERS IN CONSECUTIVE MEMORY
00359                          *  LOCATIONS FOR DMA COMPARE TEST
00360                          *
00361A F1C7 8E   6400     A BDTXFR LDX    #BYTEST  SET UP POINTER TO RAM
00362A F1CA 4F                    CLRA             CLEAR ACC
00363A F1CB A7   84       A STLP   STA    0,X      LOAD RAM WITH 0 - 256
00364A F1CD 4C                    INCA             DERIVE CONSECUTIVE #'S FOR RAM
00365A F1CE 30   01                INX              POINT TO NEXT RAM ADDRESS
00366A F1D0 8C   6500     A       CMPX    #RDTEST  CHECK FOR BYTE #255
00367A F1D3 26   F6    F1CB       BNE     STLP     IF RAM FULL CONTINUE IF NOT?
00368                          *
00369                          *
00370                          *
00371                          * WRITE A SEC THEN DMA INTO MEM THEN COMP MEM
00372                          *
00373                          *
00374                          *
00375                          *
00376                          *
00377A F1D5 BD   F000     A       JSR     INIT     INITIALIZE HARD DISK
00378A F1D8 8E   0000     A       LDX     #$0000
00379A F1DB BD   F130     A       JSR     WTSEC    WRITE CONSECUTIVE #' TO HARD DISK
00380A F1DE CE   8000     A       LDU     #DMAC
00381A F1E1 8E   0000     A       LDX     #$0000
00382A F1E4 BD   F011     A       JSR     DMA      INITIALIZE DMAC
00383A F1E7 8E   0000     A       LDX     #$0000
00384A F1EA BD   F1A8     A       JSR     DREAD    ALLOW DMAC TO READ A SECTOR
00385A F1ED 3C   EF       A       CWAI    #$EF     WAIT FOR DMA TO FINISH
00386A F1EF 8E   0000     A       LDX     #$0000
00387A F1F2 BD   F1A8     A       JSR     DREAD    ALLOW DMAC TO READ SECTOR AGAIN
00388A F1F5 8E   0000     A       LDX     #$0000
00389A F1F8 BD   F033     A       JSR     COMP     MPU WILL COMPLEMENT OLD TABLE
00390A F1FB 3C   EF       A       CWAI    #$EF     WAIT FOR DMA TO FINISH
00391A F1FD 6E   9F FFFE A       JMP     [$FFFE]  JUMP BACK TO ASSIST09
00392                            END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000


   F0F6 ACKNOW 00182 00198 00211 00228*00299 00336
   0000 ADH    00006*00056
   0002 BCH    00007*00058
   F1C7 BDTXFR 00361*
   6400 BYTEST 00025*00291 00301 00361
   0010 CHC    00008*00060 00090
   F0EB CHKCD  00174 00192 00218*00293 00328
   F107 CHKIO  00176 00239*
   3800 CNTL   00001*00040 00045 00047 00102 00111 00122 00136 00153 00157 00203
               00219 00230 00233 00239
   6700 COMND  00017*00255 00258 00261 00266 00280 00282 00285 00288 00316 00318
               00321 00324 00346 00347 00349 00351
   F033 COMP   00072*00077 00389
   3C00 DATA   00013*00039 00151 00158 00181 00195 00208 00298 00333
   0016 DCC    00011*00065
   F011 DMA    00054*00093 00382
   8000 DMAC   00012*00092 00380
```

```
---


PAGE  011  TDMA6    .SA:1

    4000 DMAT    00004*00054 00072 00074
    F1A8 DREAD   00344*00384 00387
    F10F FORMAT  00253*
    0015 ICC     00010*00063
    F000 INIT    00038*00377
    F044 ISR     00003 00087*
    3400 LATCH   00005*00066 00089
    6712 MAXD    00020*00171 00184
    0014 PRC     00009*00062
    F192 RDLOOP  00332*00338
    F16C RDSEC   00313*
    6500 RDTEST  00031*00326 00337 00366
    F18D REDATA  00328*00329
    6710 SAVEX   00019*00169 00172
    6714 SAVEX1  00021*
    F083 SELECT  00149*00253 00277 00313 00344
    F0C5 STAMSG  00087 00192*00193 00268 00304 00340
    670F STATUS  00018*00196 00209
    F0D7 STLOOP  00203*00206
    F1CB STLP    00363*00367
    F09B TXFCMD  00169*00267 00289 00325 00352
    F0A5 TXLOOP  00174*00175 00185
    F0AA TXLP1   00176*00177
    F060 WBUSY   00111*00113 00155
    F156 WDLOOP  00296*00302
    F059 WNBUSY  00102*00104 00149
    F075 WNREQ   00135*00231
    F077 WNREQ1  00136*00141
    F151 WRDATA  00293*00294
    F067 WREQ    00121*00178 00179 00194 00207 00296 00332
    F069 WREQ1   00122*00127
    F130 WTSEC   00277*00379
```

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and Ⓜ are registered trademarks of Motorola, Inc.

See Figure 11, System Schematic, inside this foldout.

**MOTOROLA LTD.** ***Semiconductor Products Sector***

COLVILLES ROAD, KELVIN ESTATE, EAST KILBRIDE, GLASGOW, G75 0TG, SCOTLAND

Printed in Great Britain by Grosvenor, Northampton. 6/85. 3000